PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L 29/06, H04N 7/52

A1

(11) International Publication Number: **WO 98/39891**

(43) International Publication Date: 11 September 1998 (11.09.98)

(21) International Application Number: PCT/US98/04459

(22) International Filing Date: 6 March 1998 (06.03.98)

(30) Priority Data:
08/813,151 7 March 1997 (07.03.97) US

(71) Applicant: MICROSOFT CORPORATION [US/US]; One Microsoft Way, Redmond, WA 98052-6399 (US).

(72) Inventors: LEVI, Steven, P.; 16612 N.E. 41st Street, Redmond, WA 98052 (US). VANANTWERP, Mark, D.; 22018 N.E. 56th Street, Redmond, WA 98053 (US). DOWELL, Craig, M.; 2429 - 234th Court N.E., Redmond, WA 98053 (US). KNOWLTON, Chadd, B.; 1430 - 169th Place N.E., Bellevue, WA 98008 (US).

(74) Agent: VIKSNINS, Ann, S.; Schwegman, Lundberg, Woessner & Kluth, P.O. Box 2938, Minneapolis, MN 55402 (US).

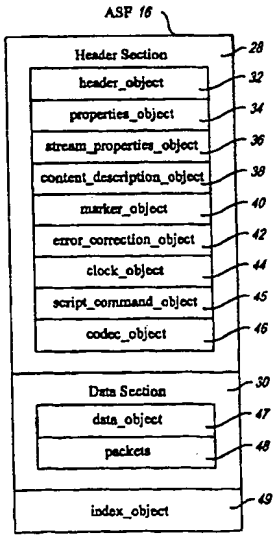
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: ACTIVE STREAM FORMAT FOR HOLDING MULTIPLE MEDIA STREAMS

(57) Abstract

An active stream format is defined and adopted for a logical structure that encapsulates multiple data streams. The data streams may be of different media. The data of the data streams is partitioned into packets that are suitable for transmission over a transport medium. The packets may include error correcting information. The packets may also include clock licenses for dictating the advancement of a clock when the data streams are rendered. The format of ASF facilitates flexibility and choice of packet size and in specifying maximum bit rate at which data may be rendered. Error concealment strategies may be employed in the packetization of data to distribute portions of samples to multiple packets. Property information may be replicated and stored in separate packets to enhance its error tolerance. The format facilitates dynamic definition of media types and the packetization of data in such dynamically defined data types within the format.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

ACTIVE STREAM FORMAT FOR HOLDING MULTIPLE MEDIA STREAMS

TECHNICAL FIELD

The present invention relates generally to data processing systems and more particularly to an active stream format for holding multiple media streams.

BACKGROUND OF THE INVENTION

Conventional file and/or stream formats for transmitting multiple data streams of varying media are limited in several respects. First, these formats are generally limited in the packet sizes that are available for encapsulating data. Such formats, if they specify packets, specify the packets as a given fixed size. Another limitation of such formats is that they do not facilitate the use of error correction codes. A further weakness of these conventional formats is that they do not provide flexibility in timing models for rendering the data encapsulated within the format. An additional limitation with such formats is that they are not well adapted for different transport mediums that have different levels of reliability and different transmission capabilities.

SUMMARY OF THE INVENTION

In accordance with a first aspect of the present invention, a computer system has a logical structure for encapsulating multiple streams of data that are partitioned into packets for holding samples of data from the multiple data streams. A method of incorporating error correction into the logical structure is performed on the computer system. In accordance with this method, a portion of at least one packet is designated for holding error correcting data. The error correcting data is then stored in the designated portion of the packet.

In accordance with another aspect of the present invention, multiple streams of data are stored in packets and error correcting data is stored in at least some of the packets. The packets are encapsulated into a larger stream and information regarding what error correcting methods are employed for the packets is also stored in the packets.

In accordance with yet another aspect of the present invention, samples of data from multiple data streams are stored in packets, and replicas of information are stored in at least some of the packets. A flag is set in each of the packets that holds replicas to indicate that the packets hold the replicas. The packets are encapsulated into a larger logical structure and transmitted to a destination.

In accordance with a further aspect of the present invention, a logical structure is provided for encapsulating multiple streams of data where the streams of data are stored in packets. Clock licenses that dictate advancement of a clock are stored in multiple ones of the packets. The logical structure is transmitted from a source computer to a destination computer. The clock is advanced at the destination computer as dictated by the clock license for each packet that holds a clock license in response to the receipt or processing of the packet at the destination computer.

In accordance with an additional aspect of the present invention, a stream format is provided for encapsulating multiple streams of data. The stream format includes a field for specifying a packet size for holding samples of the multiple streams of data. In a logical structure that adopts the stream format, a value is stored in the field that corresponds to the desired packet size. Packets of the desired size are stored within the logical structure and the logical structure is transmitted over a transport medium to the destination.

In accordance with a further aspect of the present invention, a stream format is provided for encapsulating multiple streams of data. A field is included in a logical structure that adopts the stream format for holding a value that specifies a maximum bit rate at which the multiple streams may be rendered at the destination. A value is stored in the field and the logical structure is transmitted over a transport medium to a destination.

In accordance with another aspect of the present invention, a stream format is provided for encapsulating multiple data streams and a new media type is dynamically defined. An identifier of the media type is stored in a logical structure that adopts the stream format and packets of the new media type are stored in the logical structure.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram illustrating a computer system.

Figure 2 is a flowchart illustrating use of the ASF stream.

Figure 3 is a block diagram illustrating the components of the ASF stream.

Figure 4 is a block diagram illustrating the format of the header_object.

Figure 5 is a block diagram illustrating the format of the properties_object.

Figure 6A is a flowchart illustrating the steps that are performed to fill in packet size fields within the ASF stream.

Figure 6B is a diagram illustrating different packet sizes and respective ASF streams.

Figure 7 is a block diagram illustrating the format of the stream_properties_object.

Figure 8 is a diagram that illustrates the partitioning of a sample for storage in multiple packets.

Figure 9 is a diagram that illustrates the format of the content_description_object.

Figure 10A is a diagram illustrating the format of the marker_object.

Figure 10B is a diagram illustrating the format of a marker entry.

Figure 11 is a diagram illustrating the format of the error_correction_object.

Figure 12 is flowchart illustrating the steps that are performed to utilize error correcting information.

Figure 13 is a diagram illustrating format of the clock_object.

Figure 14A is a diagram illustrating the format of the script_command_object.

Figure 14B is a diagram illustrating the format of a type_names_struc.

Figure 14C is a diagram illustrating the format of a command_entry.

Figure 15A is a diagram illustrating the format of the codec_object.

Figure 15B is a diagram of a CodecEntry.

Figure 16 is a diagram illustrating the format of the data_object.

Figure 17 illustrates the format of a packet.

Figure 18A illustrates a first format that the initial_structure may assume.

Figure 18B illustrates a second format that the initial_structure may assume.

Figure 19 illustrates the format of a payload_struc.

Figure 20 is a diagram illustrating the format of the index_object.

DETAILED DESCRIPTION OF THE INVENTION

The preferred embodiment of the present invention employs an active stream format (ASF) for holding multiple media streams. ASF is well suited for storage of multimedia streams as well as transmission of multiple media streams over a transport medium. ASF is constructed to encapsulate diverse multimedia streams and facilitates optimal interleaving of respective media streams. ASF specifies the packetization of data and provides flexibility in choosing packet sizes. In addition, ASF enables the specification of a maximum data transmission rate. As such, the packetization and transmission of media streams may be tailored to facilitate the bandwidth limitations of the system on which media streams are stored or transmitted.

ASF facilitates the use of error correction and error concealment techniques on the media streams. In unreliable transport mediums, such error correction and error concealment is highly beneficial. ASF is independent of media types and is extensible to handle newly defined media types. ASF supports flexible timing approaches and allows an author of an ASF stream to specify the

synchronization of events. ASF supports synchronized rendering using a variety of synchronization clock types and provides index information which can be used as markers for lookup to provide playback features such as fast forward and fast reverse.

Figure 1 is a block diagram of an illustrative system for practicing the preferred embodiment of the present invention. Figure 2 is a flowchart that illustrates the steps that are performed in the illustrative embodiment of Figure 1. An ASF stream 16 is built by an author (step 20 in Figure 2) and stored on a storage 14 on a source computer 10. As will be described in more detail below, ASF allows the author to design the stream for a most efficient storage based on the type of source computer 10 on which it is stored. Sometime later, the ASF stream 16 is transferred over a transport media 17, such as a network connection, to a destination computer 12 (step 24 in Figure 2). The destination computer 12 includes a number of renderers 18 for rendering the media types that are present within the ASF stream 16. For example, the ASF stream 16 may include audio-type data and video-type data. The renderers 18 at the destination 12 include an audio renderer and a video renderer. The renderers may begin rendering data as soon as they receive data prior to the complete transmission of the entire ASF stream 16 (see step 26 in Figure 2). The renderers need not immediately render the data, but rather may render the data at a later point in time.

Figure 3 depicts the basic logical organization of an ASF stream 16. It is up to the author to fill in the contents of the ASF stream in accordance with this format. The ASF stream 16 is divisible into a header section 28, a data section 30 and an index section 49. In general, the header section is first transmitted from the source computer 10 to the destination computer 12 so that the destination computer may process the information within the header section. Subsequently, the data section 30 is transmitted from the source computer 10 to the destination computer 12 on a packet-by-packet basis and the index section 49 is transmitted. The header section 28 includes a number of objects that describe the ASF stream 16 in aggregate. The header section 28 includes a header_object 32 that identifies the beginning of the ASF header section 28 and specifies the number of objects contained within the header section. Figure 4 depicts the format of the header_object 32 in more detail. The header_object 32 includes an object_id field 50 that holds a UUID for the header_object. The UUID is an identifier.

The header_object 32 also includes a size field 52 that specifies a 64-bit quantity that describes the size of the header section 28 in bytes. The header_object 32 additionally includes a number_headers field 54 that holds a 32-bit number that specifies a count of the objects contained within the header section that follow the header_object 32. An alignment field 55 specifies packing alignment of objects within the header (*e.g.*, byte alignment or word alignment). The architecture field 57 identifies the computer architecture type of the data section 30 at the index section 49. The architecture field 57 specifies the architecture of these sections as little endian or big endian.

The header_object 32 is followed in the header section 28 by a properties_object 34, such as depicted in Figure 5. The properties_object 34 describes properties about the ASF stream 16. As can be seen in Figure 5, the properties_object 34 includes an object_id field 56 that holds a UUID and a size field 58 that specifies the size of the properties_object 34. The properties_object 34 also includes a multimedia_stream_id field 60 that contains a UUID that identifies a multimedia ASF stream. A total_size field 62 is included in the properties_object 34 to hold a 64-bit value that expresses the size of the entire ASF multimedia stream.

The properties_object 34 also holds a created field 64 that holds a timestamp that specifies when the ASF stream was created. A num_packet field 65 holds a 64-bit value that defines the number of packets in the data section 30. A play_duration field 66 holds a 32-bit number that specifies the play duration of the entire ASF stream in 100-nanosecond units. For example, if the ASF stream 16 holds a movie, the duration field 66 may hold the duration of the movie. The play_duration field 66 is followed by a send_duration field 67 that corresponds to send the ASF stream in 100-nanosecond units. A preroll field 68 specifies the amount of time to buffer data before starting to play, and the flags field 70 holds 32-bits of bit flags.

The properties_object 34 includes a min_packet_size field 72 and a max_packet_size field 74. These fields 72 and 74 specify the size of the smallest and largest packets 48 in the data section 30, respectively. These fields help to determine if the ASF stream 16 is playable from servers that are constrained by packet size. For constant bit rate streams, these values are set to have the same values. A

maximum_bit_rate field 76 holds a value that specifies the maximum instantaneous bit rate (in bits per second) of the ASF stream.

Figure 6A is a flowchart illustrating how these values are identified and assigned during authoring of the ASF stream 16. First, the size of the smallest packet in the data section 30 is identified (step 78 in Figure 6A). The size of the smallest packet is stored in the min_packet_size field 72 (step 80 in Figure 6A). The size of the largest packet in the data section 30 is identified (step 82 in Figure 6A), and the size is assigned to the max_packet_size field 74 (step 84 in Figure 6A).

One of the beneficial features of ASF is its ability for facilitating different packet sizes for data of multiple media streams. Figure 6B shows one example of two different streams 83 and 85. In stream 83, each of the packets is chosen to have a size of 512 bytes, whereas in stream 85 each of the packets 48 holds 256 bytes. The decision as to the size of the packets may be influenced by the speed of the transport mechanism over which the ASF stream is to be transmitted, the protocol adopted by the transport medium, and the reliability of the transport medium.

As mentioned above, the properties_object 34 holds a value in the maximum_bit_rate field 76 that specifies an instantaneous maximum bit rate in bits per second that is required to play the ASF stream 16. The inclusion of this field 76 helps to identify the requirements necessary to play the ASF stream 16.

The header section 28 (Figure 3) must also include at least one stream_properties_object 36. The stream_properties_object 36 is associated with a particular type of media stream that is encapsulated within the ASF stream 16. For example, one of the stream_properties_objects 36 in the header section 28 may be associated with an audio stream, while another such object is associated with a video stream. Figure 7 depicts a format for such stream_properties_objects 36. Each stream_properties_object 36 includes an object_id field 86 for holding a UUID for the object and a size field 88 for holding a value that specifies the size of the object in bytes. A stream_type field 90 holds a value that identifies the media type of the associated stream.

The stream_properties_object 36 holds at least three fields 92, 98 and 104 for holding information relating to error concealment strategies. In general, ASF

facilitates the use of error concealment strategies that seek to reduce the effect of losing information regarding a given sample of media data. An example of an error concealment strategy is depicted in Figure 8. A sample 106 is divided into four sections S_1, S_2, S_3 and S_4. When the sample is incorporated into packets in the ASF stream, the samples are distributed into separate packets P_1, P_2, P_3 and P_4 so that if any of the packets are lost, the amount of data that is lost relative to the sample is not as great, and techniques, such as interpolation, may be applied to conceal the error. Each sample has a number of associated properties that describe how big the sample is, how the sample should be presented to a viewer, and what the sample holds. Since the loss of the property information could prevent the reconstruction of the sample, the properties information for the entire sample is incorporated with the portions of the sample in the packets.

The error_concealment_strategy field 92 holds a UUID that identifies the error concealment strategy that is employed by the associated stream. The error_concealment_len field 98 describes the number of bytes in an error concealment data block that is held in the error_concealment_data entries 104. The properties associated with the error concealment strategy are placed in the error_concealment_data entries 104. The number of entries will vary depending upon the error concealment strategy that is adopted.

The stream_properties_object 36 includes a stream_number field 100 that holds an alias to a stream instance. The stream_properties_object 36 also includes an offset field 94 that holds an offset value to the stream in milliseconds. This value is added to all of the timestamps of the samples in the associated stream to account for the offset of the stream with respect to the timeline of the program that renders the stream. Lastly, the stream_properties_object 36 holds a type_specific_len field 96 that holds a value that describes the number of bytes in the type_specific_data entries 102. The type_specific_data entries 102 hold properties values that are associated with the stream type.

The header section 28 (Figure 3) may also include a number of optional objects 38, 40, 42, 44, 45 and 46. These optional objects include a content_description_object 38 that holds information such as the title, author, copyright

information, and ratings information regarding the ASF stream. This information may be useful and necessary in instances wherein the ASF stream 16 is a movie or other artistic work. The content_description_object 38 includes an object_id field 110 and a size field 112 like the other objects in the header section 28. A title_len field 114 specifies the size in bytes of the title entries 119 that hold character data for the title of the ASF stream 16. An author_len field 115 specifies the size in bytes of the author entries 120 which hold the characters that specify the author of the ASF stream 16. The copyright_len field 116 holds the value that specifies the length in bytes of the copyright entries 121 that hold copyright information regarding the ASF stream 16. The description_len field 117 holds a value that specifies the length in bytes of the description entries 122. The description entries 122 hold a narrative description of the ASF stream 16. Lastly, the rating_len field 118 specifies a size in bytes of the rating entries 123 that hold rating information (*e.g.*, X, R, PG-13) for the ASF stream content.

The header section 28 may include a marker_object 40. The marker_object 40 holds a pointer to a specific time within the data section 30. The marker_object enables a user to quickly jump forward or backward to specific data points (*e.g.*, audio tracks) that are designated by markers held within the marker_object 40.

Figure 10A shows the marker_object 40 in more detail. The marker_object 40 includes an object_id field 126 that holds a UUID, and a size field 128 specifies the size of the marker_object in bytes. A marker_id field 130 contains a UUID that identifies the marker data strategy, and a num_entries field 132 specifies the number of marker entries in the marker_object 40. An entry_alignment field 134 identifies the byte alignment of the marker data, and a name_len field 136 specifies how many Unicode characters are held in the name field 138, which holds the name of the marker_object 40. Lastly, the marker_data field 140 holds the markers in a table. Each marker has an associated entry in the table.

Figure 10B shows the format of a marker entry 141 such as found in the marker_data field 140. An offset field 142 holds an offset in bytes from the start of packets in the data_object 47 indicating the position of the marker entry 141. A time field 144 specifies a time stamp for the marker entry 141. An entry_len field 146

specifies the size of an entry_data field 148, which is an array holding the data for the marker entry.

The header section 28 may also include an error_correction_object 42 for an error correction method that is employed in the ASF stream. Up to four error correction methods may be defined for the ASF stream 16 and, thus, up to four error_correction_objects 42 may be stored within the header section 28 of the ASF stream 16. Figure 11 depicts the format of the error_correction_object 42.

The error_correction_object 42 includes an object_id field 150 and a size field 152, like those described above for the other objects in the header section 28. The error_correction_object 42 also includes an error_correction_id 154 that holds UUID that identifies the error correcting methodology associated with the object 42. The error_correction_data_len field 156 specifies the length in bytes of the error_correction_data entries 158 that hold octets for error correction. The error_correction_object 42 is used by the destination computer 12 (Figure 1) in playing the ASF stream 16.

Figure 12 depicts a flowchart of how error correcting may be applied in the preferred embodiment of the present invention. In particular, an error correction methodology such as an N+1 parity scheme, is applied to one or more streams within the ASF stream 16 (step 160 in Figure 12). Information regarding the error correcting methodology is then stored in the error_correction_object 42 within the header section 28 (step 162 in Figure 12). The source computer then accesses the error correcting methodology information stored in the error_correction_object 42 in playing back the ASF stream 16 (step 164 in Figure 12). Error correcting data is stored in the interleave_packets 48.

The header section 28 of the ASF stream 16 may also hold a clock_object 44 that defines properties for the timeline for which events are synchronized and against which multimedia objects are presented. Figure 13 depicts the format of the clock_object 44. An object_ID field 166 holds a UUID to identify the object, and a size field 168 identifies the size of the clock_object 44 in bytes. A packet_clock_type field 170 identifies the UUID of the clock_type that is used by the object. A packet_clock_size field 172 identifies the clock size. A clock_specific_len

field 174 identifies the size and bytes of the clock_specific_data field 176 which contains clock-specific data. The clock type alternatives include a clock that has a 32-bit source value and a 16-bit duration value, a clock type that has a 64-bit source value and a 32-bit duration value and a clock type that has a 64-bit source value and a 64-bit duration value.

The ASF stream 16 enables script commands to be embedded as a table in the script_command_object 45. This object 45 may be found in the header section 28 of the ASF stream 16. The script commands ride the ASF stream 16 to the client where they are grabbed by event handlers and executed. Figure 14A illustrates the format of the script_command_object 45. Like many of the other objects in the header section 28, this object 45 may include an object_ID field 178 for holding a UUID for the object and a size field 180 for holding the size in bytes of the object. A command_ID field 182 identifies the structure of the command entry that is held within the object.

The num_commands field 184 specifies the total number of script commands that are to be executed. The num_types field 186 specifies the total number of different types of script_command types that have been specified. The type_names field 188 is an array of type_names_struc data structures. Figure 14B depicts the format of this data structure 192. The type_name_len field 194 specifies the number of Unicode characters in the type_names field 196, which is a Unicode string array holding names that specify script command types.

The command_entry field 190 identifies what commands should be executed at which point in the timeline. The command_entry field 190 is implemented as a table of script commands. Each command has an associated command_entry element 198 as shown in Figure 14C. Each such element 198 has a time field 200 that specifies when the script command is to be executed and a type field 202 that is an index into the type_names array 196 that identifies the start of a Unicode string for the command type. A parameter field 204 holds a parameter value for the script command type.

The script commands may be of a URL type that causes a client browser to be executed to display an indicated URL. The script command may also be of a file

name type that launches another ASF file to facilitate "continuous play" audio or video presentations. Those skilled in the art will appreciate that other types of script commands may also be used.

The header section 28 of the ASF stream 16 may also include a codec_object 46. The codec_object 46 provides a mechanism to embed information about a codec dependency that is needed to render the data stream by that codec. The codec object includes a list of codec types (*e.g.*, ACM or ICM) and a descriptive name which enables the construction of a codec property page on the client. Figure 15A depicts the format of a codec_object 46. The object_id field 206 holds a UUID for the codec_object 46 and the size field 208 specifies the size of the object 46 in bytes. The codec_ID field 210 holds a UUID that specifies the codec_ type used by the object. The codec_entry_len field 212 specifies the number of CodecEntry entries that are in the codec_entry field 214. The codec_entry field 214 contains codec-specific data and is an array of CodecEntry elements.

Figure 15B depicts the format of a single CodecEntry element 216 as found in the codec_entry field 214. A type field 218 specifies the type of codec. A name field 222 holds an array of Unicode characters that specifies the name of the codec and a name_len field 220 specifies the number of Unicode characters in the name field. The description field 226 holds a description of the codec in Unicode characters and the description_len field 224 specifies the number of Unicode characters held within the description field. The cbinfo field 230 holds an array of octets that identify the type of the codec and the cbinfo_len field 228 holds the number of bytes in the cbinfo field 230.

As mentioned above, the data section 30 follows the header section 28 in the ASF stream 16. The data section includes a data_object 47 and interleave_packets 48. A data_object 47 marks the beginning of the data section 30 and correlates the header section 28 with the data section 30. The packets 48 hold the data payloads for the media stream stored within the ASF stream 16.

Figure 16 depicts the format of the data_object 46. Like other objects in the ASF stream 16, data_object 46 includes an object_id field 232 and a size field 234. The data_object 46 also includes a multimedia_stream_id field 236 that holds a UUID

for the ASF stream 16. This value must match the value held in the multimedia_stream_id field 60 in the properties_object 34 in the header section 28. The data_object 46 also includes a num_packets field 238 that specifies the number of interleave_packets 48 in the data section 30. An alignment field 240 specifies the packing alignment within packets (*e.g.*, byte alignment or word alignment), and the packet_alignment field 242 specifies the packet packing alignment.

Each packet 48 has a format like that depicted in Figure 17. Each packet 48 begins with an initial_structure 244. The format of the initial_structures 244 depends upon whether the first bit held within the structure is set or not. Figure 18A depicts a first format of the initial_structure 244 when the most significant bit is cleared (*i.e.*, has a value of zero). The most significant bit is the error_correction_present flag 270 that specifies whether error correction information is present within the initial_structure 244 or not. In this case, because the bit 270 is cleared, there is no error correction information contained within the initial_structure 244. This bit indicates whether or not error correction is used within the packet. The two bits that constitute the packet_len_type field 272 specify the size of the packet_len field 256, which will be described in more detail below. The next two bits constitute the padding_len_type field 274 and specify the length of the padding_len field 260, which will also be discussed in more detail below. The next two bits constitute the sequence_type field 276 and specify the size of the sequence field 258. The final bit is the multiple_payloads_present flag 278 which specifies whether or not multiple payloads are present within the packet. A value of 1 indicates that multiple media stream samples (*i.e.*, multiple payloads) are present within the packet.

Figure 18B depicts the format of the initial_structure 244 when the error_correction_present bit is set (*i.e.*, has a value of 1). In this instance, the first byte of the initial_structure 244 constitutes the ec_flag field 280. The first bit within the ec_flag field is the error_correction_present bit 270, which has been described above. The two bits that follow the error_correction_present bit 270 constitute the error_correction_len_type field 284 and specify the size of the error_correction_data_len field 290. The next bit constitutes the opaque_data flag 286 which specifies whether opaque data exists or not. The final four bits constitute the

error_correction_data_length field 288. If the error_correction_len_type field 284 has a value of "00" then the error_correction_data_length field 288 holds the error_correction_data_len value and the error_correction_data_len field 290 does not exist. Otherwise this field 288 has a value of "0000." When the error_correction_data_len field 290 is present, it specifies the number of bytes in the error_correction_data array 292. The error_correction_data array 292 holds an array of bytes that contain the actual per-packet data required to implement the selected error correction method.

The initial_structure 244 may also include opaque data 300 if the opaque_data bit 286 is set. The initial structure includes a byte of flags 302. The most significant bit is a reserved bit 304 that is set to a value of "0." The next two bits constitute the packet_len_type field 306 that indicate the size of the packet_len field 256. The next subsequent two bits constitute the padding_len_type field 272 that indicate the size of the padding_len field 274. These two bits are followed by another 2-bit field that constitutes the sequence_type of field 276 that specifies the size of the sequence field 258. The last bit is the multiple_payloads_present bit 278 that specifies whether are not multiple payloads are present.

The initial_structure 244 is followed by a stream_flag field 246 that holds a byte consisting of four 2-bit fields. The first two bits constitute a stream_id_type field 248 that specifies the size of the stream_id field 314 within the payload_struc 266. The second most significant bits constitute the object_id_type field 250 and indicate the number of bits in the object_id field 316 of the payload_struc 266 as either 0-bits, 8-bits, 16-bits or 32-bits. The third most significant two bits constitute the offset_type field 252, which specifies the length of the offset field 318 within the payload_struc 266 as either 0-bits, 8-bits, 16-bits or 32-bits. The least two significant bits constitute the replicated_data_type field 254 and these bits indicate the number of bits that are present for the replicated_data_len field 320 of the payload_struc 266.

The packet 48 also includes a packet_len field 256 that specifies the packet length size. The sequence field 258 specifies the sequence number for the packet. The padding_len field 260 contains a number that specifies the number of

padding bytes that are present at the end of the packet to pad out the packet to a desirable size.

The packet 48 also contains a clock_data field 262 that contains data representing time information. This data may include a clock license that contains a system clock reference that drives the progression of the time line under the timing model and a duration that specifies the effective duration of the clock license. The duration field limits the validity of the license to a time specified in milliseconds. Under the model adopted by the preferred embodiment of the present invention, the source computer 10 issues a clock license to the destination computer 12 that allows the clock of the destination computer 12 to progress forward for a period of time. The progression of time is gated by the arrival of a new piece of data that contains a clock value with a valid clock license that is not expired.

The packet 48 also includes a payload_flag field 264 that specifies a payload length type and a designation of the number of payloads present in the packet. The payload_flag field 264 is followed by one or more payload_strucs 266. These structures contain payload information which will be described in more detail below. The final bits within the packet 48 may constitute padding 268.

Figure 19 depicts the payload_struc 266 in more detail. The stream_id field 314 is an optional field that identifies the stream type of the payload. The object_id field 316 may be included to hold an object identifier. An offset field 318 may be included to specify an offset of the payload within the ASF stream. The offset represents the starting address within a zero-address-based media stream sample where the packet payload should be copied.

The payload_struc 266 may also include a replicated_data_len field 320 that specifies the number of bytes of replicated data present in the replicated_data field 322. As was discussed above, for protection against possible errors, the packet 48 may include replicated data. This replicated data is stored within the replicated_data field 322.

The payload_len field 323 specifies the number of payload bytes present in the payload held within the payload_data field 325. The payload_data field 326 holds an array of payloads (*i.e.*, the data).

The ASF stream may also include an index_object 49 that holds index information regarding the ASF stream 16. Figure 20 depicts the format of the index_object 49. The index_object includes a number of index entries. The index_object 49 includes an object_id field 324 and a size field 326. In addition, the index_object 49 includes an index_id field 328 that holds a UUID for the index type. Multiple index_name_entries may be stored depending on the number of entries required to hold the characters of the name. For example, each entry may hold 16 characters in an illustrative embodiment.

The index_object includes a time_delta field 330 that specifies a time interval between index entries. The time represents a point on the timeline for the ASF stream 16. A max_packets field 332 specifies a maximum value for packet_count fields, which will be described in more detail below. A num_entries field 334 is a 32-bit unsigned integer that describes the maximum number of index entries that are defined within the index_info array 336. This array 336 is an array of index_information structures. Each index_info structure holds a packet field that holds a packet number associated with the index entry and a packet_count field specifies the number of the packet to send with the index entry so as to associate the index entries with the packets. In Figure 21, the index_info array structure 336 holds N index_information structures and each index_information structure has a packet field 338A-338N and a packet_count field 340A-340N.

While the present invention has been described with reference to a preferred embodiment thereof, those skilled in the art will appreciate that various changes in form and detail may be made without departing from the intended scope of the invention as defined in the appended claims. For example, the present invention may be practiced with a stream format that differs from the format described above. The particulars described above are intended merely to be illustrative. The present invention may be practiced with stream formats that include only a subset of the above-described fields or include additional fields that differ from those described above. Moreover, the length of the values held within the fields and the organization of the structures described above are not intended to limit the scope of the present invention.

CLAIMS

1. In a computer system having a logical structure for encapsulating multiple streams of data that are partitioned into packets for holding samples of data from the multiple data streams, a method of incorporating error correction into the logical structure, comprising the steps of:

designating a portion of at least one packet for holding error correcting data; and

storing error correcting data in the designated portion of at least one packet.

2. The method of claim 1, further comprising the step of using the error correcting data stored in the designated portion of at least one packet to correct an error.

3. The method of claim 1 wherein the error correcting data is stored in multiple packets.

4. The method of claim 1 wherein the error correcting data holds parity bits.

5. The method of claim 1 wherein packets holding samples of data from a first of the multiple streams holds a different type of error correcting data than packets holding samples of data from a second of the multiple streams.

6. The method of claim 1 wherein the logical structure includes a header that holds information regarding what error correcting methodology is used with the at least one packet that holds error correcting data.

7. The method of claim 1, further comprising the step of:

transferring the packets across a transport medium to a destination computer.

8. The method of claim 1 wherein at least two of the multiple streams of data are of different media.

9. In a computer system, a method comprising the steps of:

storing data from multiple streams of data into packets;

storing error correcting data in at least some of the packets;

encapsulating the packets into a larger stream; and

storing information regarding what error correcting methods are employed for the packets in the larger stream.

10. The method of clam 9 wherein the larger stream includes a header and the information regarding the error correcting methods that are employed is stored in the header.

11. The method of claim 10 wherein multiple error correcting methods are employed.

12. The method of claim 11 wherein a separate object is stored in the header for each error correcting method to encapsulate the information regarding the error correcting method.

13. In a computer system that is coupled to a destination a method comprising the steps of:

storing samples of data from multiple data streams in packets;

storing replicas of information in at least some of the packets;

setting a flag in the packets that hold the replicas to indicate that the packets hold replicas;

encapsulating the packets into a larger logical structure; and

transmitting the logical structure to the destination.

14. The method of claim 13 wherein the replicas of information hold property information regarding the samples of data.

15. The method of claim 13 wherein portions of a sample are stored in selected packets and a replica of property information regarding the sample is stored in each packet in which a portion of the sample is stored.

16. The method of claim 13, further comprising the step of examining one of the replicas of information at the destination when one of the packets is lost during the transmitting.

17. In a computer system having a source computer and a destination computer having a clock that regulates timing of activities at the destination computer, a method comprising the steps of:

providing a logical structure for encapsulating multiple streams of data, said streams of data being stored in packets;

storing clock licenses that dictate advancement of a clock in multiple ones of the packets;

transmitting the logical structure from the source computer to the destination computer; and

for each packet that holds a clock license, advancing the clock at the destination computer as dictated by the clock license in response to receiving the packet at the destination computer.

18. The method of claim 17 wherein each clock license includes a time value to which the clock at the destination computer is to be advanced.

19. The method of claim 18 wherein each clock license includes an expiration time after which the clock license is invalid.

20. In a distributed system having a computer system that is coupled to a destination via a transport medium, a method comprising the steps of:

providing a stream format for encapsulating multiple streams of data;

including a field in a logical structure that adopts the stream format for specifying a packet size for holding samples of the multiple streams of data;

storing a value in the field that corresponds with a desired packet size;

storing packets of the desired size within the logical structure that adopts the stream format; and

transmitting the logical structure over the transport medium to the destination.

21. The method of claim 1 wherein the logical structure that adopts the stream format holds a field for a maximum packet size and a field for a minimum packet size.

22. The method of claim 20 wherein at least two of the multiple streams of data hold data of different media.

23. In a distributed system having a computer system that is connected with a destination via a transport medium, a method comprising the steps of:

providing a stream format for encapsulating multiple streams of data;

including a field in a logical structure that adopts the stream format for holding a value that specifies a maximum bit rate at which the multiple streams of data may be rendered at the destination;

storing a value in the field; and

transmitting the logical structure over the transport medium to the destination.

24. In a data processing system, a method comprising the steps of:

providing a stream format for encapsulating multiple data streams;

dynamically defining a new media type;

storing an identifier of the new media type in a logical structure that adopts the stream format; and

storing packets of data of the new media type in the logical structure.

25. The method of claim 24 wherein the data processing system includes a transport medium and a target computer, further comprising the step of sending the logical structure over the transport medium to the target computer.

26. The method of claim 24 wherein a renderer for the new media type is stored at the target computer, wherein the logical structure includes a field that identifies a renderer and wherein the method further comprises the step of accessing the field in the logical structure that identifies the renderer at the target computer to determine what renderer to use to render data of new media type.

27. In a computer system having a logical structure for encapsulating multiple streams of data that are partitioned into packets for holding samples of data from the multiple streams of data, a computer-readable storage medium holding instructions for:

designating a portion of at least one packet for holding error correcting data; and

storing error correcting data in the designated portion of at least one packet.

28. The computer-readable storage medium of claim 27 holding instructions that store the error correcting data in multiple packets.

29. The computer-readable storage medium of claim 27 wherein the medium holds instructions for encapsulating a first type of error correcting data for packets that hold samples from a first of the streams of data and encapsulating a second type of error correcting data for packets that hold samples from a second of the streams of data.

30. In a computer system, a computer-readable storage medium holding a logical structure that encapsulates multiple streams of data, said logical structure comprising:

packets of data from the multiple streams of data for transmission over a transport medium; and

error correcting data within at least some of the packets at designated locations.

31. The computer-readable storage medium of claim 30 wherein the logical structure further comprises a header in which information regarding an error correcting method that uses the error correcting data is stored.

32. The computer-readable storage medium of claim 31 wherein the header holds information regarding multiple error correcting methods.

33. The computer-readable storage medium of claim 30 wherein the multiple streams of data include at least two streams of different media.

34. In a computer system, a computer-readable storage medium holding instructions for:

receiving a logical structure that holds multiple streams of data wherein said streams of data include samples that are stored in packets in the logical structure and wherein at least some of the packets include error correcting data; and

extracting the error correcting data from at least some of the packets as needed to correct errors.

35. In a computer system, a computer-readable storage medium holding a logical structure that includes:

samples of data from multiple data streams in packets;

replicas of information in at least some of the packets; and

a flag in the packets that holds the replicas to indicate that the packets hold replicas.

36. The computer-readable storage medium of claim 35 wherein portions of a sample are stored in selected packets and a replica of property information regarding the sample is stored in each packet in which a portion of the sample is stored.

37. In a computer system, a computer-readable storage medium holding a logical structure that encapsulates:

multiple streams of data wherein the streams of data are stored in packets;

clock licenses that each dictate advancement of a clock that regulates rendering of the data in the packets.

38. The computer-readable storage medium of claim 37 wherein each clock license includes a time value to which the clock at the destination computer is to be advanced.

39. The computer-readable storage medium of claim 38 wherein each clock license includes an expiration time after which the clock license is invalid.

40. In a computer system that is coupled to a destination computer via a transport mechanism, a computer-readable storage medium holding a logical structure comprising:

multiple streams of data; and

a value in a field that specifies a maximum bit rate at which the multiple streams of data may be rendered at the destination computer.

41. A data processing system having:

a source computer with a storage;

a logical structure stored in storage for encapsulating multiple data streams, data from said data streams being incorporated in packets;

error correcting data encapsulated in the packets.

42. A data processing system having:

a source computer with a storage;

a logical structure stored in storage for encapsulating multiple data streams, data from said data streams being incorporated in packets;

a clock license being encapsulated into at least one packet for advancing a clock at a destination when processed at the destination.

43. A method of encapsulating a plurality of diverse data streams into a formatted data stream comprising:

partitioning samples from each data stream into type-specific packets;

interleaving the type-specific packets for the plurality of diverse data streams to create a single data stream; and

adding a stream header into the single data stream to create a formatted data stream, wherein the stream header comprises information regarding the format of the formatted data stream and the packets in the formatted data stream.

44. A computer-readable medium having computer-executable instructions for performing the steps of:

partitioning samples from each data stream into type-specific packets;

interleaving the type-specific packets for the plurality of diverse data streams to create a single data stream; and

adding a stream header into the single data stream to create a formatted data stream, wherein the stream header comprises information regarding the format of the formatted data stream and the packets in the formatted data stream.

45. In a computer system that is coupled to a destination, a method comprising the steps of:

partitioning samples from each data stream into type-specific packets;

interleaving the type-specific packets for the plurality of diverse data streams to create a single data stream;

adding a stream header into the single data stream to create a formatted data stream, wherein the stream header comprises information regarding the format of the formatted data stream and the packets in the formatted data stream; and

transmitting the formatted data stream to the destination.

46. In data processing system having:

a source computer with a storage; and

a logical structure stored in storage for encapsulating diverse data streams into a formatted data stream, wherein the logical structure comprises interleaved packets containing samples from the diverse data streams, and a stream header comprising information regarding the format of the formatted data stream and the packets in the formatted data stream.

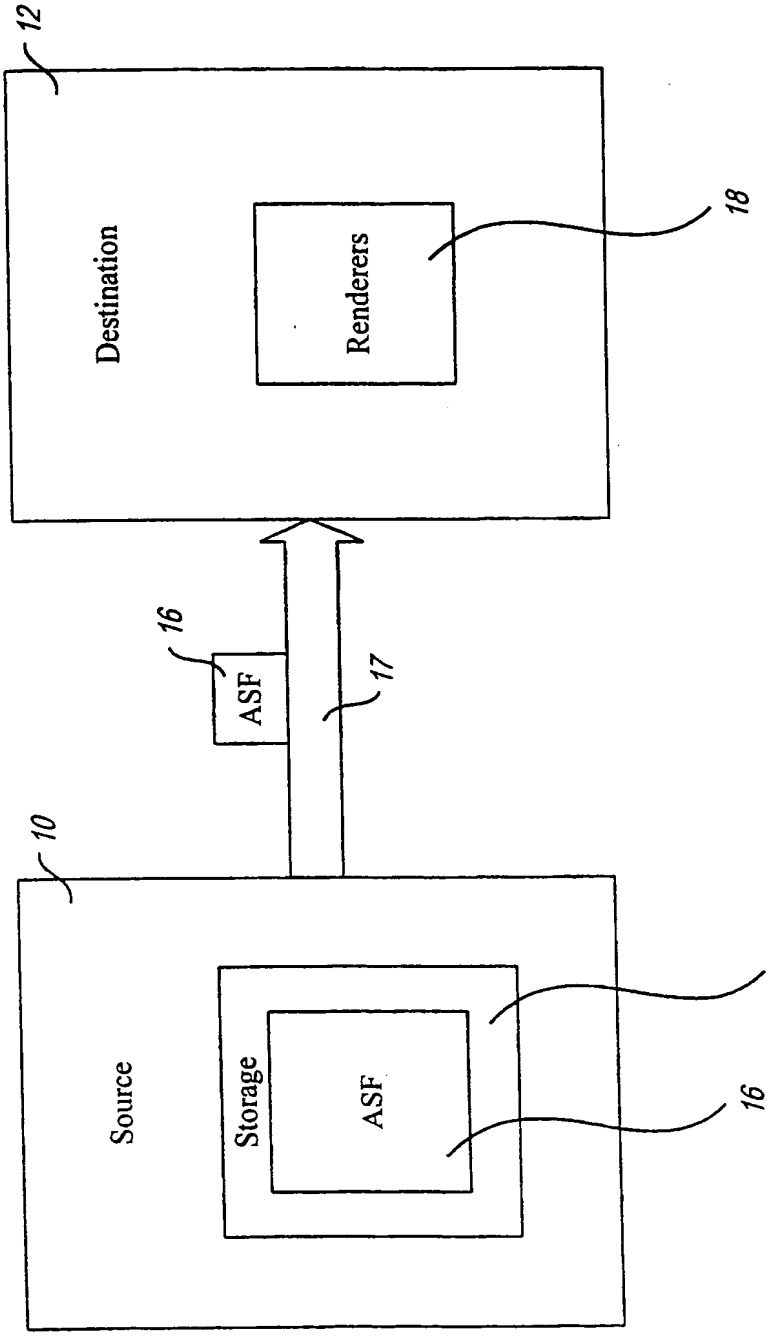

12
Destination
Renderers
18
16
ASF
17
10
Source
Storage
ASF
14
16

Fig. 1

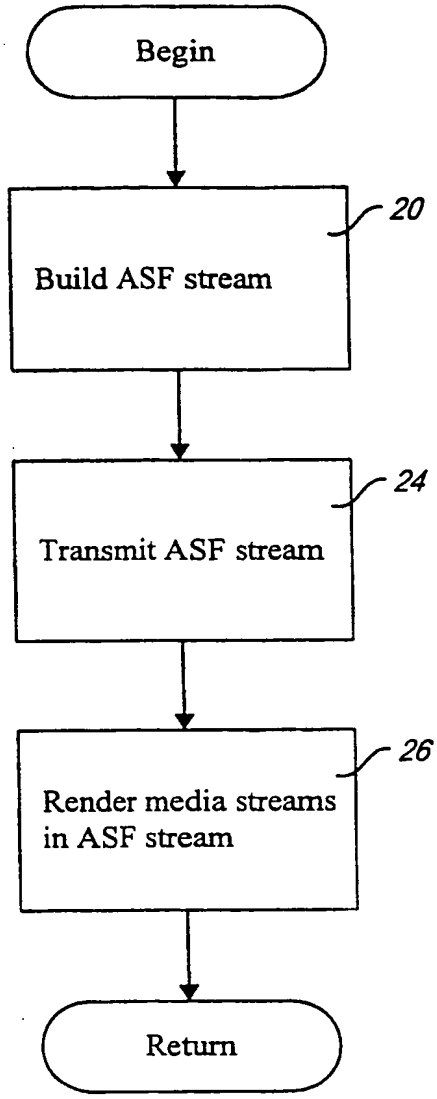

Begin
Build ASF stream
20
Transmit ASF stream
24
Render media streams in ASF stream
26
Return

Fig. 2

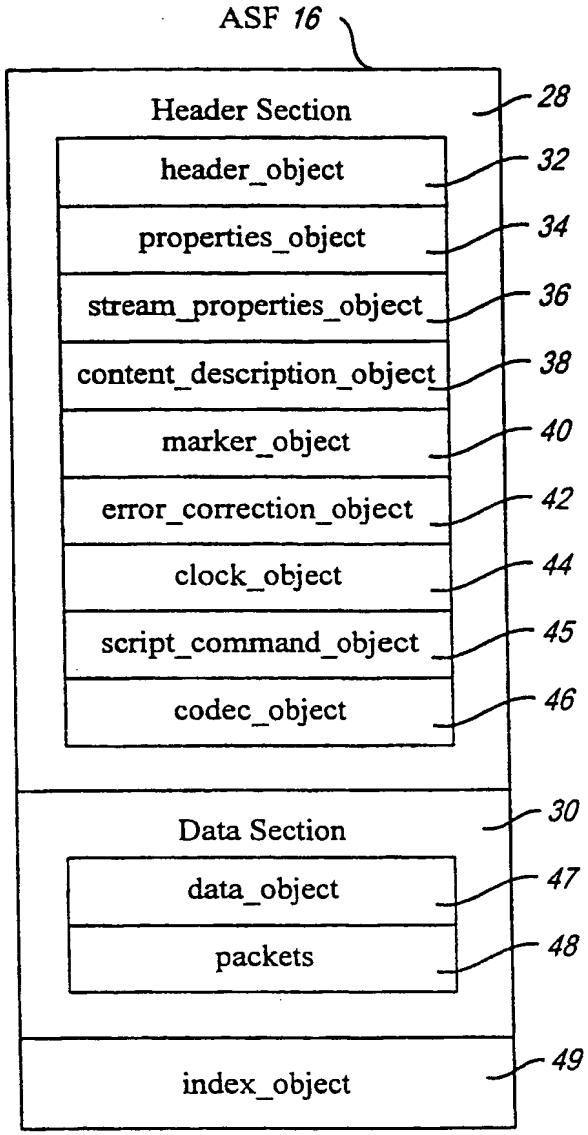

ASF 16
Header Section
28
header_object
32
properties_object
34
stream_properties_object
36
content_description_object
38
marker_object
40
error_correction_object
42
clock_object
44
script_command_object
45
codec_object
46
Data Section
30
data_object
47
packets
48
index_object
49

Fig. 3

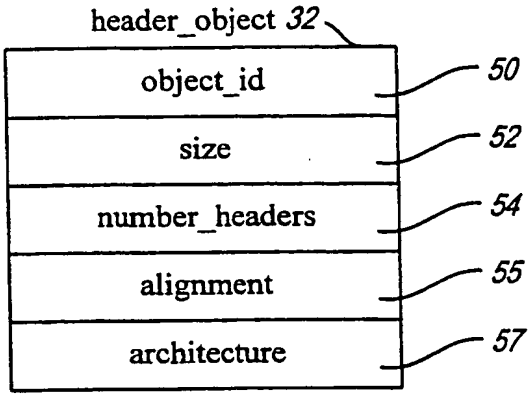

header_object 32
object_id
50
size
52
number_headers
54
alignment
55
architecture
57

Fig. 4

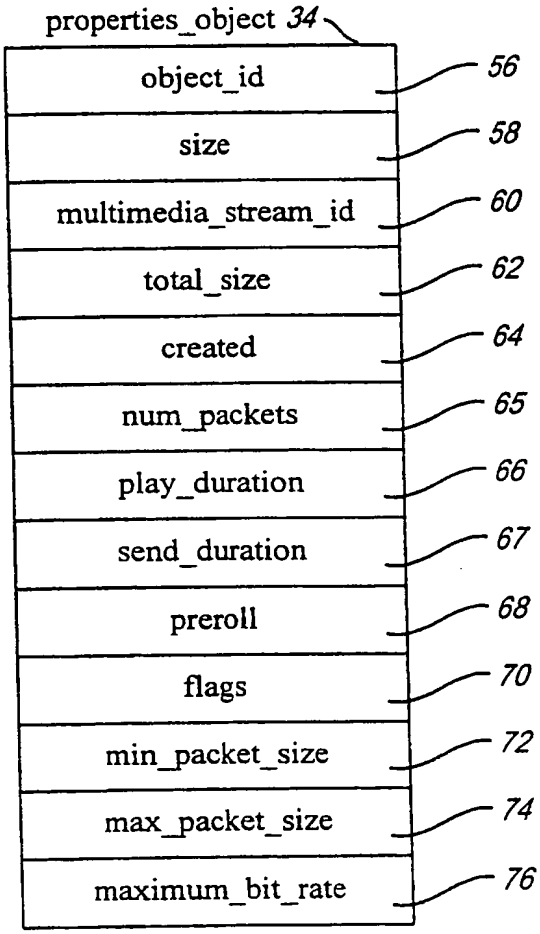

properties_object 34
object_id
56
size
58
multimedia_stream_id
60
total_size
62
created
64
num_packets
65
play_duration
66
send_duration
67
preroll
68
flags
70
min_packet_size
72
max_packet_size
74
maximum_bit_rate
76

Fig. 5

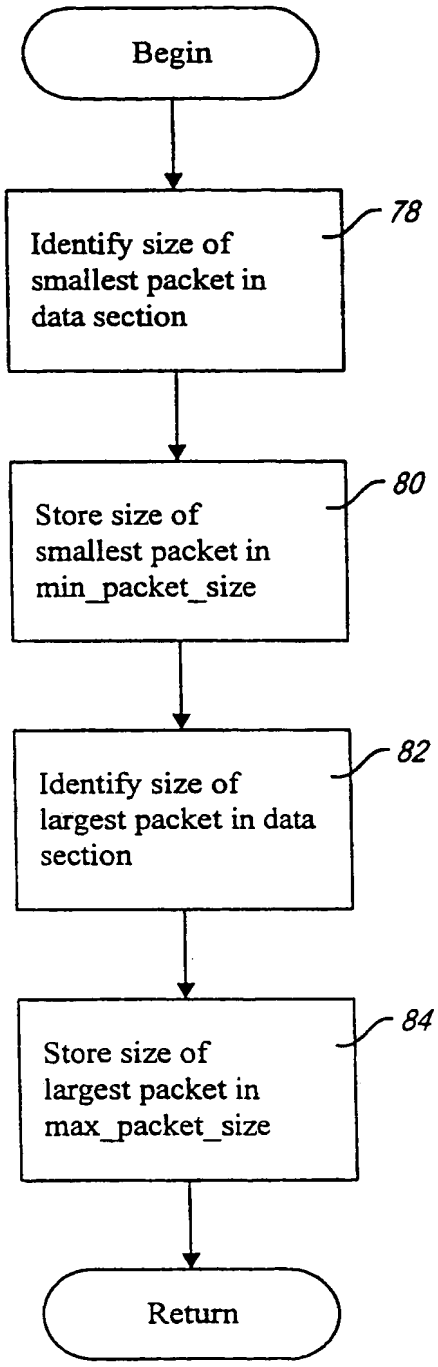

Begin
Identify size of smallest packet in data section
78
Store size of smallest packet in min_packet_size
80
Identify size of largest packet in data section
82
Store size of largest packet in max_packet_size
84
Return

Fig. 6A

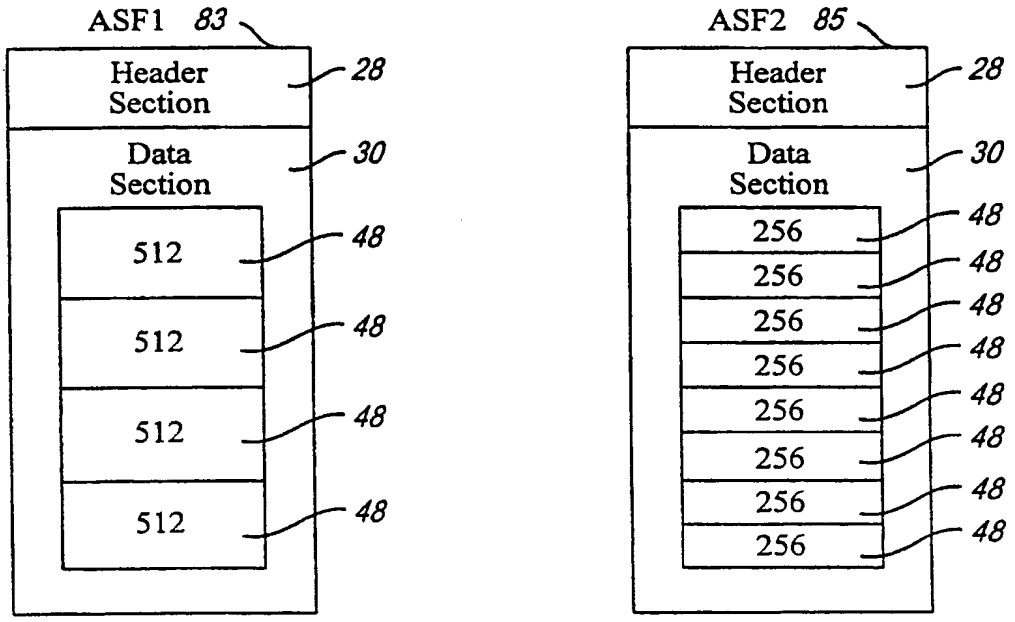

ASF1 83
Header Section
28
Data Section
30
512
48
512
48
512
48
512
48
ASF2 85
Header Section
28
Data Section
30
256
48
256
48
256
48
256
48
256
48
256
48
256
48
256
48

Fig. 6B

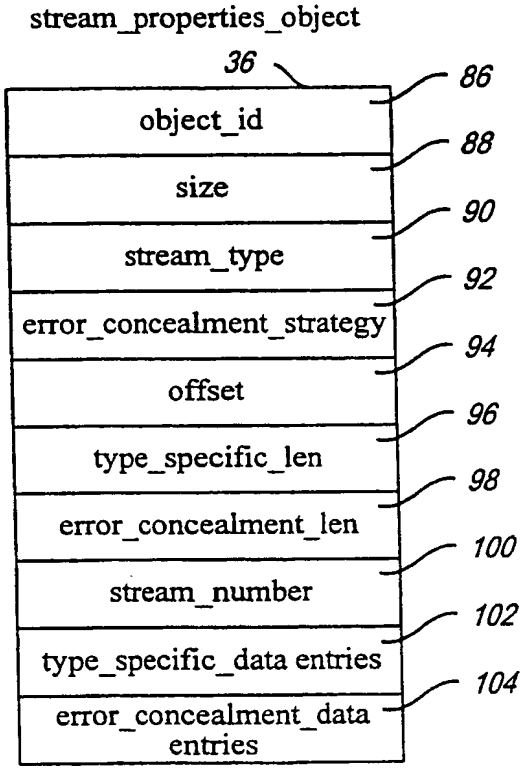

stream_properties_object
36
object_id
86
size
88
stream_type
90
error_concealment_strategy
92
offset
94
type_specific_len
96
error_concealment_len
98
stream_number
100
type_specific_data entries
102
error_concealment_data entries
104

Fig. 7

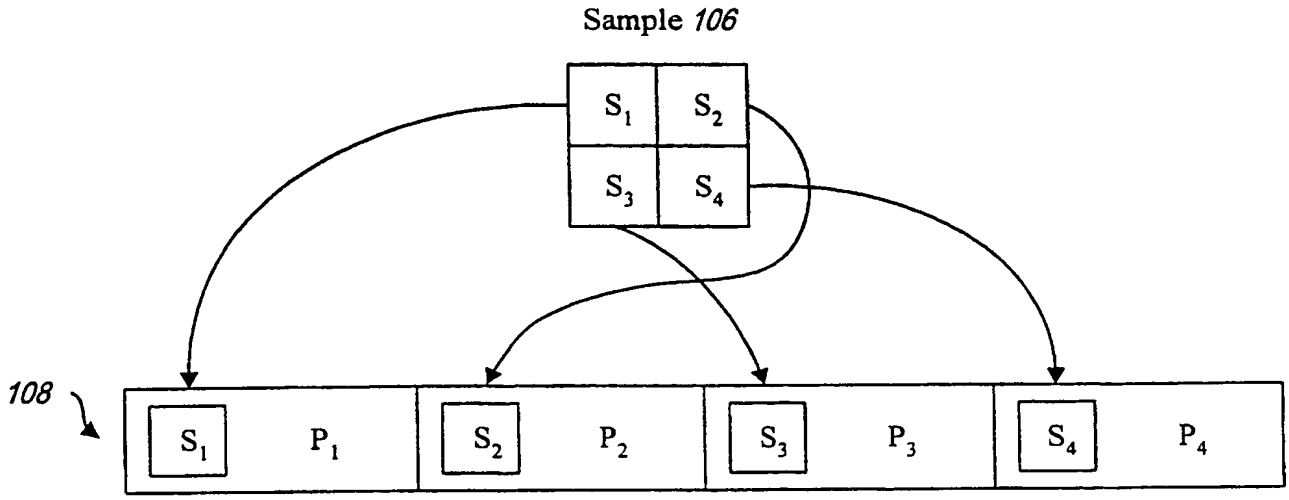


Fig. 8

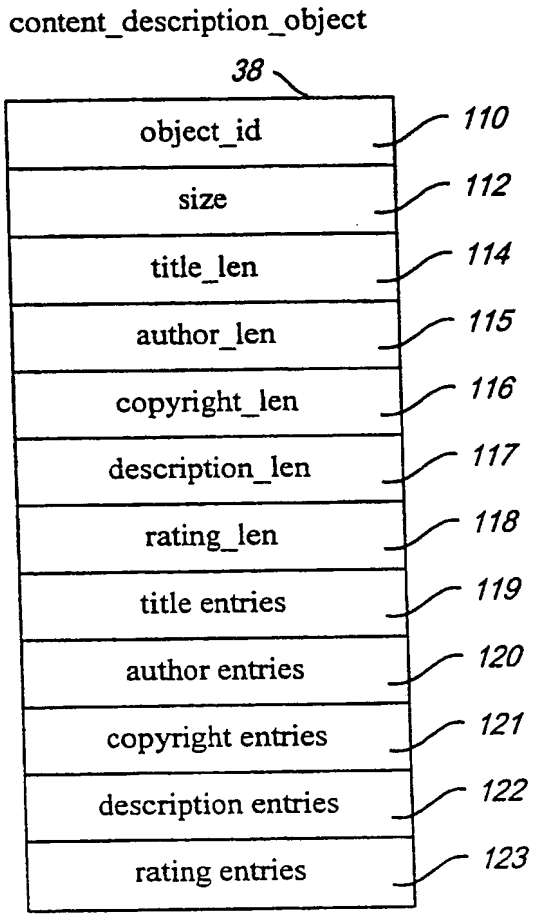

content_description_object
38
object_id
110
size
112
title_len
114
author_len
115
copyright_len
116
description_len
117
rating_len
118
title entries
119
author entries
120
copyright entries
121
description entries
122
rating entries
123

Fig. 9

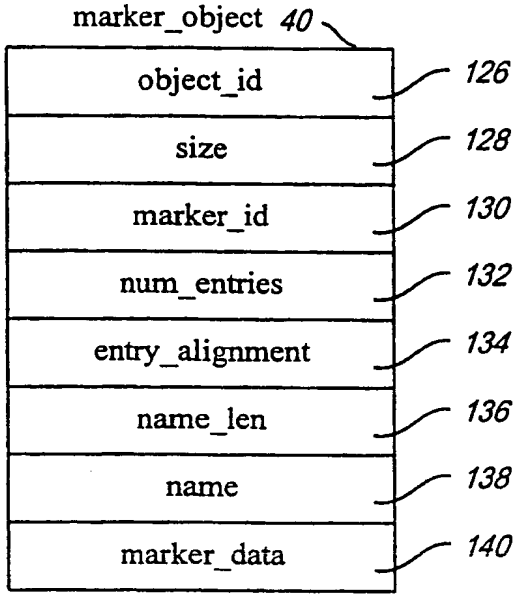

marker_object 40
object_id
126
size
128
marker_id
130
num_entries
132
entry_alignment
134
name_len
136
name
138
marker_data
140

Fig. 10A

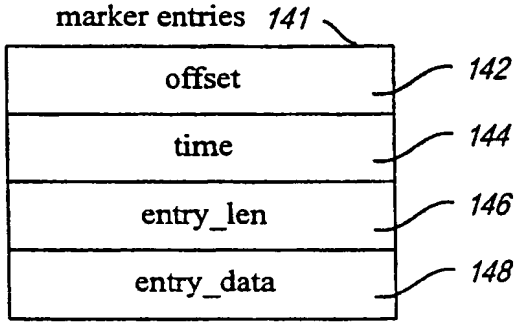

marker entries 141
offset
142
time
144
entry_len
146
entry_data
148

Fig. 10B

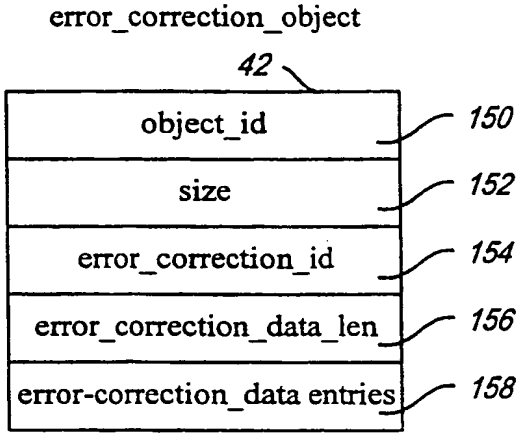

error_correction_object
42
object_id
150
size
152
error_correction_id
154
error_correction_data_len
156
error-correction_data entries
158

Fig. 11

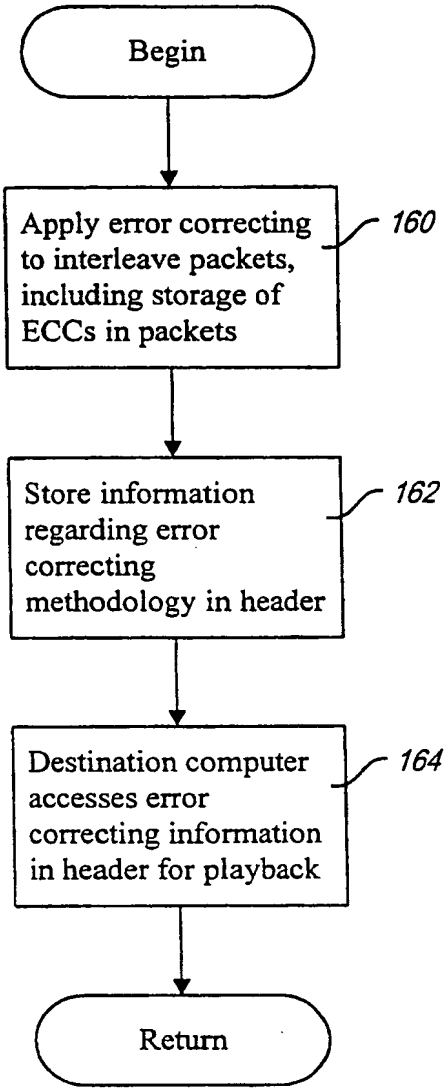

Begin
Apply error correcting to interleave packets, including storage of ECCs in packets
160
Store information regarding error correcting methodology in header
162
Destination computer accesses error correcting information in header for playback
164
Return

Fig. 12

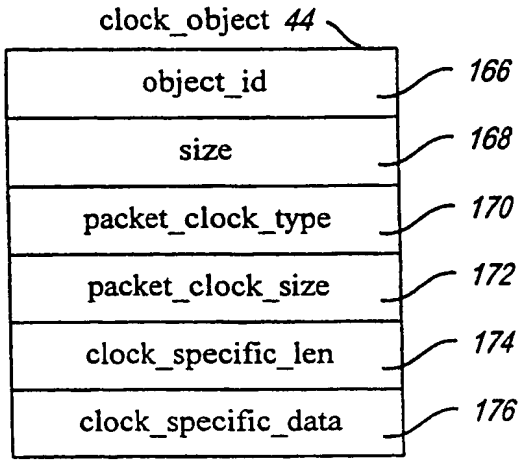

clock_object 44
object_id
166
size
168
packet_clock_type
170
packet_clock_size
172
clock_specific_len
174
clock_specific_data
176

Fig. 13

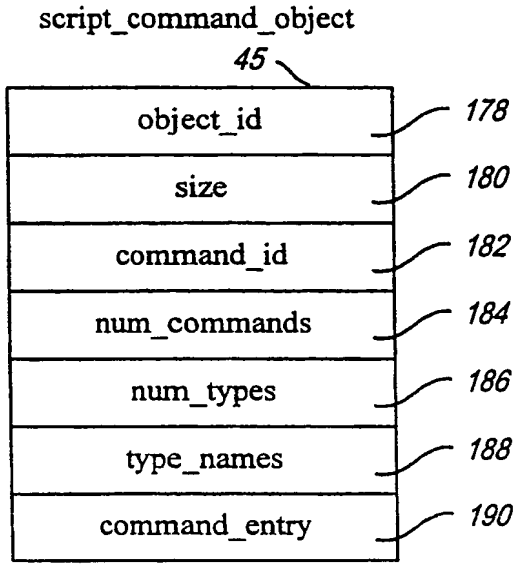

script_command_object
45
object_id
178
size
180
command_id
182
num_commands
184
num_types
186
type_names
188
command_entry
190

Fig. 14A

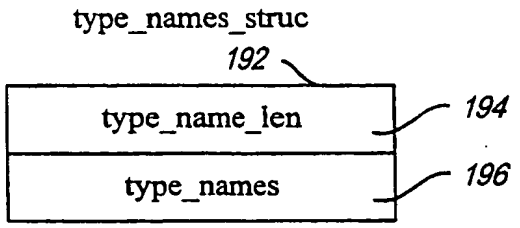


Fig. 14B

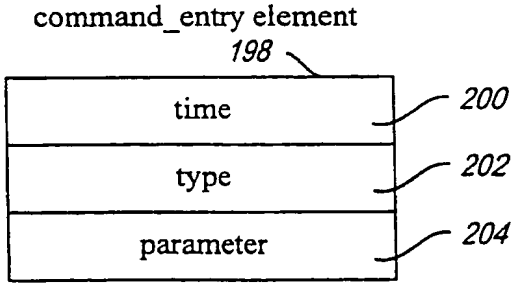


Fig. 14C

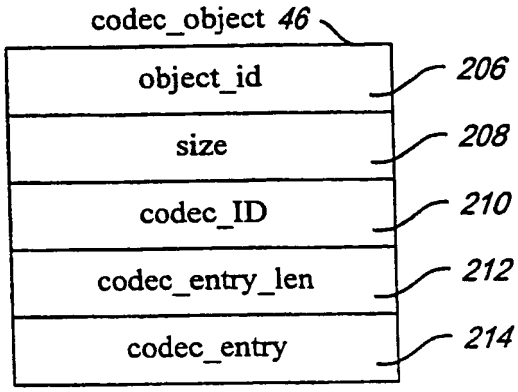

codec_object 46
object_id
206
size
208
codec_ID
210
codec_entry_len
212
codec_entry
214

Fig. 15A

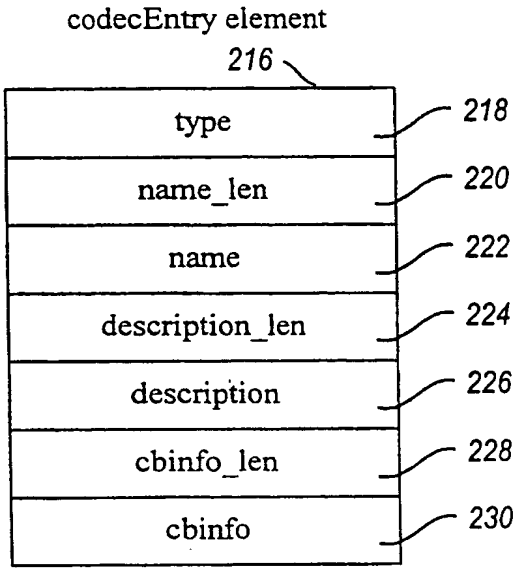

codecEntry element
216
type
218
name_len
220
name
222
description_len
224
description
226
cbinfo_len
228
cbinfo
230

Fig. 15B

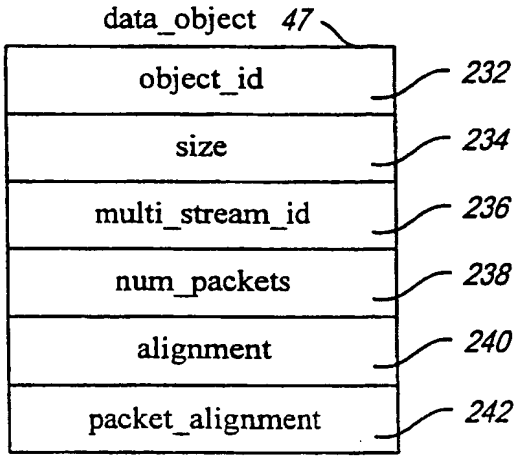

data_object 47
object_id
232
size
234
multi_stream_id
236
num_packets
238
alignment
240
packet_alignment
242

Fig. 16

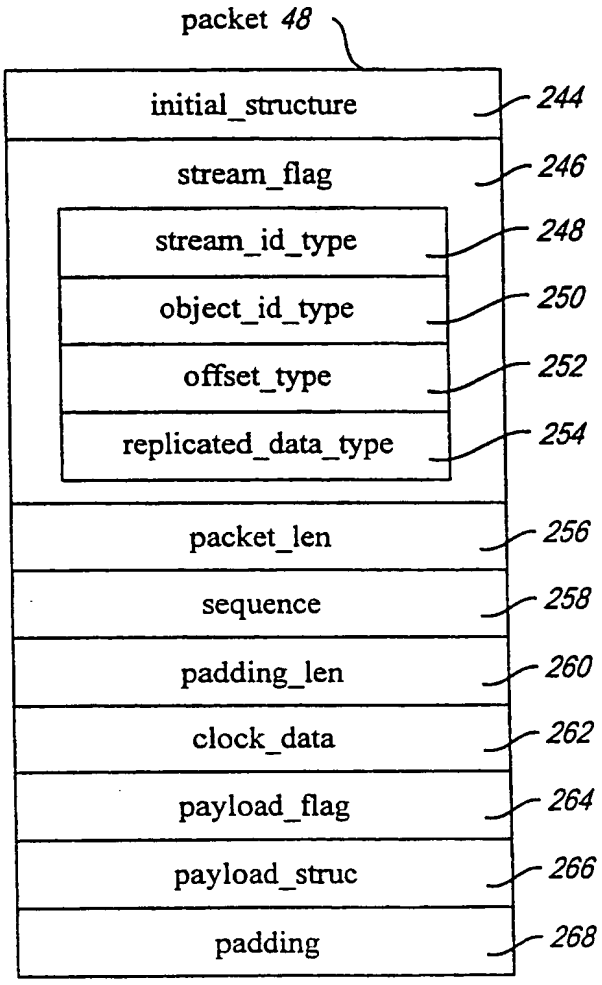

packet 48
initial_structure
244
stream_flag
246
stream_id_type
248
object_id_type
250
offset_type
252
replicated_data_type
254
packet_len
256
sequence
258
padding_len
260
clock_data
262
payload_flag
264
payload_struc
266
padding
268

Fig. 17

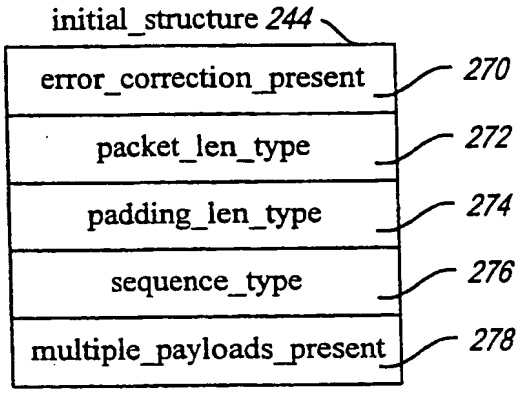

initial_structure 244
error_correction_present
270
packet_len_type
272
padding_len_type
274
sequence_type
276
multiple_payloads_present
278

Fig. 18A

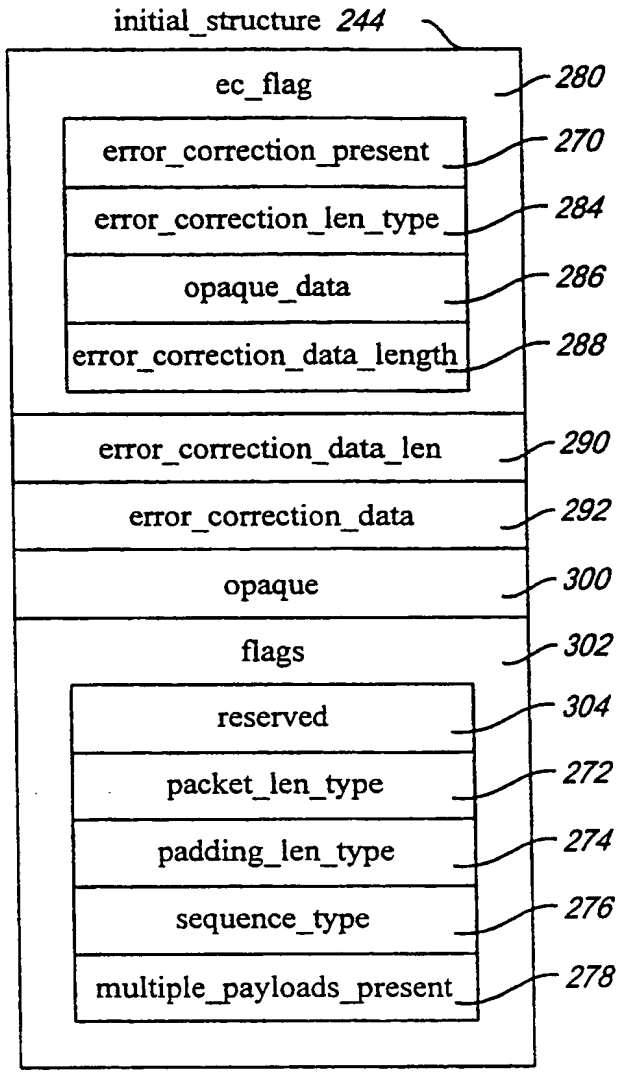

initial_structure 244
ec_flag
280
error_correction_present
270
error_correction_len_type
284
opaque_data
286
error_correction_data_length
288
error_correction_data_len
290
error_correction_data
292
opaque
300
flags
302
reserved
304
packet_len_type
272
padding_len_type
274
sequence_type
276
multiple_payloads_present
278

Fig. 18B

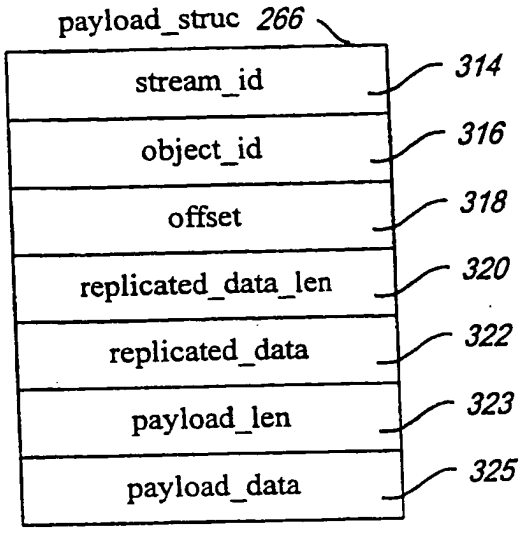

payload_struc 266
stream_id
314
object_id
316
offset
318
replicated_data_len
320
replicated_data
322
payload_len
323
payload_data
325

Fig. 19

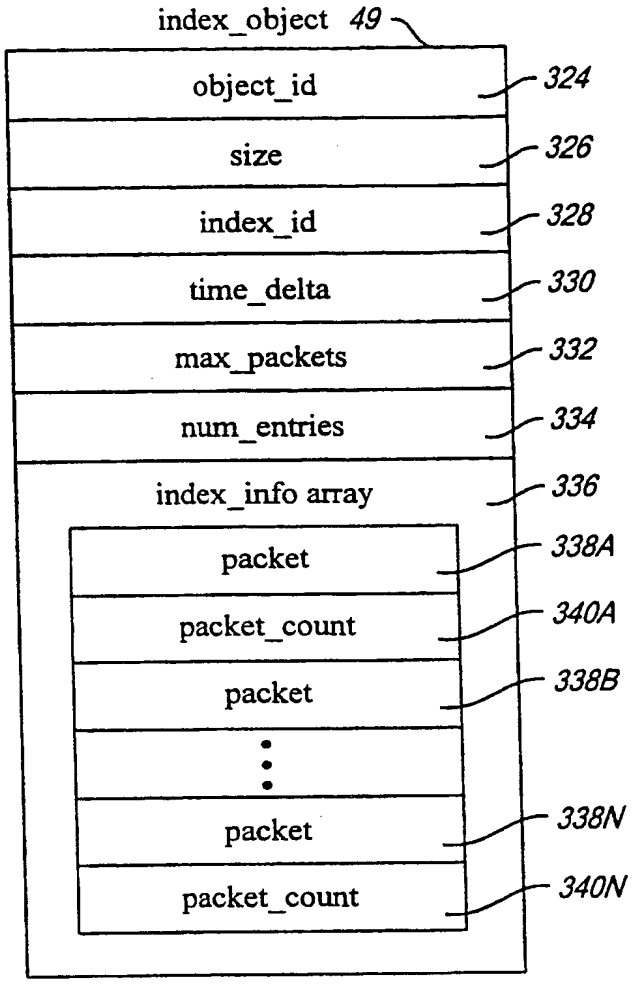

index_object 49
object_id
324
size
326
index_id
328
time_delta
330
max_packets
332
num_entries
334
index_info array
336
packet
338A
packet_count
340A
packet
338B
packet
338N
packet_count
340N

Fig. 20

INTERNATIONAL SEARCH REPORT

In ational Application No
PCT/US 98/04459

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04L29/06 H04N7/52

According to International Patent Classification(IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04L H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	JAU-SHIUNG HUANG ET AL: "MHTP - A MULTIMEDIA HIGH-SPEED TRANSPORT PROTOCOL" COMMUNICATION FOR GLOBAL USERS, ORLANDO, DEC. 6 - 9, 1992, vol. VOL. 3, no. -, 6 December 1992, INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, pages 1364-1368, XP000390432	1-4,6-9, 23-28, 30,31, 33,34, 40,41
Y	see page 1364, column 1, line 1 - page 1366, column 1, line 36	5,29,32
A	see page 1366, column 2, line 14 - line 22	10-12

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publicationdate of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of theinternational search	Date of mailing of the international search report
15 June 1998	23/06/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Karavassilis, N

2

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 98/04459

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	SARGINSON P A: "MPEG-2: A TUTORIAL INTRODUCTION TO THE SYSTEMS LAYER" IEE COLLOQUIUM ON MPEG WHAT IT IS AND WHAT IT ISN'T, 1 January 1995, pages 4/1-4/13, XP000560804	1-4,8, 17,18, 20, 22-25, 27,28, 30,33, 34,37, 38,40-46
A	see page 1, line 1 - page 3, line 8 see page 3, line 22 - page 5, line 12 see figures 3-5	10,19,39
X	LA PORTA T F ET AL: "THE MULTISTREAM PROTOCOL: A HIGHLY FLEXIBLE HIGH-SPEED TRANSPORT PROTOCOL" IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, vol. 11, no. 4, 1 May 1993, pages 519-530, XP000402610 see abstract see page 519, column 2, line 13 - page 520, column 2, line 38; figure 1 see page 522, column 1, line 14 - line 28	1-4,7,8, 20,22, 27,28, 30,33, 34,41
X	SCHTZMAYR R ET AL: "PROVIDING SUPPORT FOR DATA TRANSFER IN A NEW NETWORKING ENVIRONMENT" MULTIMEDIA TRANSPORT AND TELESERVICES. INTERNATIONAL COST 237 WORKS PROCEEDINGS, VIENNA, NOV. 13 - 15, 1994, 13 November 1994, HUTCHISON D;DANTHINE A; LEOPOLD H; COULSON G (EDS), pages 241-255, XP000585304 see abstract see page 241, line 14 - page 242, line 3 see page 242, line 40 - line 45 see page 243, line 27 - line 32; figure 1 see page 248, line 6 - line 33 see page 251, line 10 - line 13; figure 6 see page 252, line 11 - line 14; figure 8	1-4,7,8, 13-16, 27,28, 30, 33-36,41
X	EP 0 753 954 A (GEN INSTRUMENT CORP) 15 January 1997	17,18, 37,38
A	see the whole document	19,39
Y	PATENT ABSTRACTS OF JAPAN vol. 096, no. 001, 31 January 1996 & JP 07 245600 A (NIPPON TELEGR & TELEPH CORP), 19 September 1995,	5,29,32
A	see abstract	10-12

-/--

2

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 98/04459

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 604 843 A (WU ZHANBING ET AL) 18 February 1997 see column 3, line 59 - line 66 see column 4, line 38 - column 5, line 17 see claim 10	27-42, 44,46
A	OHTA K ET AL: "A PROPOSAL OF NETWORK PROTOCOL WITH PERFORMANCE FOR MULTIMEDIA COMMUNICATION SYSTEM" IEICE TRANSACTIONS ON INFORMATION AND SYSTEMS, vol. E79-D, no. 6, 1 June 1996, pages 719-727, XP000595177 see figure 5	21

INTERNATIONAL SEARCH REPORT

Information on patent family members

In ational Application No
PCT/US 98/04459

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0753954 A	15-01-1997	US 5598415 A	28-01-1997
		AU 5835496 A	23-01-1997
		CA 2181194 A	14-01-1997
		NO 962910 A	14-01-1997
		JP 9135443 A	20-05-1997
		US 5742623 A	21-04-1998
US 5604843 A	18-02-1997	NONE	